1. Locking and deadlocks (5 questions)

1) There are two phases in two-phase locking: ___ phase and ____ phase.

2) Consider the following transaction:

```
T1:                              T2:

_______                   __________

read(A);                        read(B)

_______                   __________

read(B);                        read(A)

if A = 0 then B:=B+1;        if B = 0 then A:=A+1;

write(B);                       write(A);


_______                         _______

_______                         _______
```

Please add proper lock and unlock instructions(lock/unlock S( ) or lock/unlock X( )) to transactions T1 and T2 in (1)~(4), so that they observe the strict two-phase locking protocol.

3) Can the execution of transactions T1 and T2 in 2) with the strict two-phase locking protocol result in a deadlock? If your answer is yes, please show a schedule where deadlock happens. If your answer is no, please explain why.

4) Can the execution of transactions T1 and T2 in 2) with the conservative two-phase locking protocol result in a deadlock? If your answer is yes, please show a schedule where deadlock happens. If your answer is no, please explain why.

5) For either strict two phase locking or conservative two phase locking, is there a conflict serializable schedule that is equivalent (but different) than a serial schedule that does not result in a deadlock? If your answer is yes, please show such a schedule (for either strict or conservative two phase locking). If the answer is no, please explain why (for both strict and conservative two phase locking).

Answer:

No.
For strict two phase locking, it is not possible to interleave the transactions without a deadlock. If we interleave even the first operation in one of the transactions, Once we reach the third operation the transaction will wait for the other, and then the other transaction will reach the third operation and deadlock.

For conservative two phase locking, if we run one transaction, it will grab an exclusive lock either on A or B, which will prevent the other transaction from even starting.

2.Log (3 questions)

Suppose we have an excerpt of a write-ahead log of table ***Balance*** as:

LSN	Prev LSN	Tx ID	Type	Loc	Old Val	New Val
100	99	10	UPDATE	x1.Balance	0	100
101	0	11	BEGIN	null	null	null
102	101	11	UPDATE	x2.Balance	100	90
103	100	10	COMMIT	null	null	null
104	103	11	UPDATE	x3.Balance	80	90
105	0	12	BEGIN	null	null	null
106	105	12	UPDATE	x1.Balance	100	50
107	104	11	ABORT	null	null	null

where each transaction has a unique transaction id (Tx ID) and each log entry has a log sequence number (LSN).

Assume all transactions with Tx ID below 10 has been committed and the table ***Balance*** at the time right before log 100 is:

ID	Name	Balance
x1	Rico	94
x2	Nick	103
x3	Tom	62

Answer the following questions:

1. Please show the table in memory after log 103.
2. Suppose the system flashed the memory into disk after log 103, but crashed after flashing, please show the table after recovery.
3. Please show the table in memory after log 107.

x1	Rico	100
x2	Nick	90/103
x3	Tom	62

x1	Rico	100
x2	Nick	103
x3	Tom	62

x1	Rico	50/100
x2	Nick	103
x3	Tom	62

3.SQL (1 question)

For the following question please refer to the following database schema:
Person(SSN, employerSymbol, salary)
Holding(SSN, symbol, numShares)
A person is uniquely identified by a social security number (SSN). A company is uniquely identified by its stock ticker symbol. Each person is employed by exactly one company, but may hold any number of different stocks.

Suppose we wish to find the average salary of the persons who own more than 100 shares of Microsoft (MSFT) or more than 100 shares of Tesla (TSLA). Which of the following queries will correctly compute the desired average?

I.
```
SELECT AVG(salary)
FROM Person
WHERE SSN IN (
        SELECT SSN
        FROM Holding
        WHERE (symbol = ’MSFT’ OR symbol = ‘TSLA’) AND numShares > 100);
```

II.
```
SELECT AVG(salary)
FROM Person, Holding
WHERE Person.SSN = Holding.SSN AND ((symbol = ’MSFT’ AND numShares > 100) OR
(symbol = ‘TSLA’ AND numShares > 100));
```

(A) I only
(B) II only
(C) Both I and II
(D) Neither I nor II

Answer: A
(II) is incorrect. Suppose John owns 200 shares of MSFT and 200 shares of TSLA, we end up with 2 tuples of John; therefore John's salaries will be counted twice in the average.

4. Key Value Stores (3 questions)

Redis is an in-memory (i.e., all of its data is stored in memory) key-value store, and it supports different kinds of data structures, such as strings, lists, and maps, as well as a simple key-value GET, PUT interface. It is one of the most popular key-value databases.

1. What are one advantage and two disadvantages of Redis compared to on-disk SQL databases?

Possible answers:
Advantages: 1. Lower latency, higher throughput
Disadvantages: 1. No durability, 2. No ACID support, 3. No SQL support

2. How would you solve one of the disadvantages listed above if you are a designer of Redis?

Answer:
We can flush Redis operations to disk.

3. Redis provides two ways to persist data into disks. One is *snapshotting*: it periodically (could be 1 hour, 1 day, ...) flushes the in-memory data in a compact file and store it in disk. Another is *logging*: it logs every write operation to the disk. What are the advantages and disadvantages of each approach?

Answer:
Snapshotting:
Pro: More efficient to write data sequentially to disk. Fewer writes. Higher throughput/lower latency.
Con: you may lose operations 1 hour/ 1 day ago

Logging:
Pro: can recover very recent updates.
Con: slower/more I/O than snapshotting.

5. Systems back of the envelope (7 questions)

Assume the following average read latencies:
Reading from a magnetic disk database: 10ms
Reading from a flash database: 1ms
Reading from memory: 100us

You are designing a data science application, hosted in your own data center that stores election data and predicts future elections. For now, assume that you have an infinitely fast network (with a latency of 0 for servers to talk to each other), and ignore all latencies except for the ones listed above. Compute the average latency for each of the following configurations:

1. Store the entire database on a single magnetic disk

Answer: 10 ms

2. *Shard* (or split) the database onto 10 small flash databases that can be accessed in parallel, with a single extra flash database that stores the index. The request first goes to the index server, and then to one of the 10 small flash databases.

Answer: 1 ms + 1 ms = 2 ms

3. Put a key-value in-memory store (a KV store fully in memory) that caches requests before the setup described in 2. above. When a request comes in, the key-value store has a 50% chance of being cached in memory, and a 50% that it won't exist in memory and needs to be fetched from the flash index and subsequently one of the 10 sharded databases. Round your answer to the nearest millisecond.

Answer: 0.5 * 100us + 0.5 * (100 us + 1 ms + 1 ms) ~= 1 ms
P(cached) * $cache_access + P(uncached) * ($cache_access + $index_access + $data_access)

4. The same setup as 3., but we also add a bloom filter in memory **before** the KV store. The bloom filter has a false positive rate of 1%, and 50% of the requests do not exist in the database (and the KV store). You can use the rounded answer for question 3. when calculating the solution to this question. In this question round your answer to the nearest half a millisecond.

Answer: 0.5 * (100 us + $p3_answer) + 0.5 * (0.01*(100 us + 100 us + 1 ms) + 0.99*100 us) ~= 0.5ms

P(existent) * ($bloom_filter_access + $p3_answer) + P(inexistent) * (P(false positive) * ($bloom_filter_access + $cache_access + $index_access) + P(true negative) * $bloom_filter_access)

5. Your boss doesn't just want you to optimize for average latency, but also for maximum latency (i.e., the worst possible latency a user might see). Please explain in what situations magnetic disks exhibit high maximum latency.

Answer: seeking from one end of the disk to another and/or waiting for an entire rotation.

6. Now assume that the database is hosted on Google Cloud, and that the average roundtrip time to Google Cloud over the Internet is 100ms. You can still assume the network **within** Google Cloud is infinitely fast.
   You do a user study, and the conclusions are that users cannot notice the difference in latency for requests that are under 150ms. Assume that the cost of a single magnetic disk database is $100, while the cost of each small flash partition or index database is $20, and the cost of an in-memory cache or bloom filter is $110. Which configuration would you pick among 1-4 and why?

Answer:
1. $100, $220, $330, $330 or $440
2. 110 ms is less than 150 ms, and even the maximum latency of magnetic disks would still be under 150 ms. Magnetic disks are much cheaper. (As long as the answer makes sense)

Note: Cost for strategy 4) may vary due to different interpretations whether cache and bloom filter is stored in the same memory or separately.

7. At the end your boss was convinced to go with configuration 4 hosted on Google Cloud. One day, your application suddenly starts having latencies of 200 ms. Please sort the following problems in terms of the amount of potential delay they might cause:

A. The in-memory KV store has crashed, so all requests go directly to the flash index server
B. The bloom filter has doubled its false positive rate
C. The index server's latency is 10X slower
D. The Internet connection has performance problems

Answer: D, C, A, B.

Explanation: Internet is by far the slowest -- so is most likely to be the problem. Index server sits in the path of every request, so if its latency went up 10X the total latency the total latency will be affected by several milliseconds. The in-memory KV can halve the latency to the database, so if it went down it could double the total latency of the KV. Finally a doubling of the false positive rate of the Bloom Filter is not so significant -- it may add tens of microseconds of latency, but not milliseconds.

6. Conflicts and anomalies (7 questions)

Your database needs to process the following transactions:

T1
r1(A), w1(A)

T2
r2(A), w2(B)

T3
w3(A), r3(B), r3(A)

You are given the following schedule, **S1**:

r1(A), r2(A), w3(A), w2(B), w1(A), r3(B), r3(A)

1. Please list **all** the operations conflict with each other, and what is the type of conflict:

r1(A), w3(A), RW
r2(A), w3(A), RW
r2(A), w1(A), RW
w3(A), w1(A), WW
w2(B), r3(B), WR
w1(A), r3(A), WR

2. Please draw the conflict graph of this schedule:

Answer:
1 → 3
2 → 3
2 → 1
3 → 1

3. Is this graph conflict serializable? If yes, which serial schedule is it equivalent to. If no, please explain why.

Answer: no. There is a cycle in the conflict graph between 1,2

You are given the following schedule, **S2**:

r2(A), w3(A), r1(A), w2(B), r3(B), r3(A), w1(A)

4. Please draw the conflict graph of this schedule:

Answer:
2 → 3
2 → 1
3 → 1

5. Is this graph conflict serializable? If yes, which serial schedule is it equivalent to. If no, please explain why.

Answer: Yes, T2→T3→T1

6. For both schedules S1, S2, please provide another example of a conflict serializable schedule that involves only changing the position of r1(A) (i.e., you're allowed to move around r1(A), but no other operation within the schedule). If there is no such schedule, please explain why

Answer:
For S1 there is no way to just move r1(A) and make a conflict serializable schedule, because the order of operations (w3(A)→w1(A)→r3(A) already create a bad schedule.

For S2 a possible schedules is one of the following three:
r2(A), w3(A), w2(B), r1(A), r3(B), r3(A), w1(A)
r2(A), w3(A), w2(B), r3(B), r1(A), r3(A), w1(A)
r2(A), w3(A), w2(B), r3(B), r3(A), r1(A), w1(A)

7. In our database Transaction 1 and 3 are relatively fast to execute, while in transaction 2, in between r2(A) and w2(B), the application needs to do some heavy computations on the values it read from A.
   We want to allow our database to execute other operations while the application is doing the computations for transaction 2.
   a. What is the latest point in the schedule where the database can execute w2(B) and still be conflict serializable and equivalent to S1 or S2?
   b. In order to meet this condition, which operations must come **before and after** w2(B)?

Answer:

a. The latest point is the operation before the last one (i.e., the sixth operation). For example:

   r2(A), w3(A), r1(A), r3(A), w1(A), w2(B), r3(B)

b.
   w2(B) must be executed before r3(B), since they have a WR conflict.

   w2(B) must be executed after r2(A) to respect the order of operations within the transaction.

7. Amdahl's law (2 questions)

Compute the speedup in the following scenarios (answers are rounded to the closest decimal):

1. We moved our indices from flash to memory, which sped up 50% of the queries by 1000X

I. 1000
II. 0.5
III. 2
IV. 1

Answer: 1 / (0.5 + 0.5 / 1000) ~= 1 / (0.5) = 2

2. We moved our bloom filter from flash to memory. 50% of the requests are for objects not found, but the bloom filter has a 10% false positive rate. The bloom filter speeds up object not found requests that **are not false positives** by 1000X.

I. 1.8
II. 2
III. 2.2
IV. 2.5

Answer: 1 / (0.55 + 0.45 / 1000) = 1.8

8. Bloom filter (2 questions)

1. Do regular Bloom filters support deletion? If yes, explain how the operation works. If no, explain why not.

Answer: no. If we were to reset the bits of the hash of a deleted key to 0, we might be inadvertently resetting the bits that belong to some other key that hashes to some (or all) of the bits of the deleted key. This would violate the property of a Bloom filter, where it only has false positives, but not false negatives.

2. Imagine you have a Bloom filter with a certain number of bits, m. You run a workload where new keys are constantly inserted. What would be the expected false positive rate of the Bloom filter over time?

I. Increases monotonically over time, until it becomes 100%
II. Remains constant
III. Decreases over time
IV. Fluctuates (goes up and down)

Answer: I. Bloom filters don’t support deletions, so if we infinitely insert objects eventually the FP rate will become 1. The FP rate always increases, and it can never decrease, since we don’t support deletions.